ELBUG
SUPPLEMENT TO BEEBUG

January/February 1986 - Volume 3 issue 3

MAKING THE MOST OF THE FUNCTION KEYS (Part 2)

Dave Robinson continues his study of the Electron's function keys with further hints and tips.

INVISIBLE FUNCTION KEYS

When a function key is pressed the definition will appear on the screen, just as it would if you had typed it in directly from the keyboard. Often this is not desirable. It would be preferable if the mini-program could be run without it being visible. It is possible to hide the definition in various ways:

1. Include a mode change at the end of the definition.

2. Add CLS at the end of the definition.

3. Define a zero-sized window for the definition to 'appear' in.

If the first method is chosen, you can implement the mode change in two formats:

```
*KEY8 -- definition -- MO.6|M

*KEY8 -- definition -- |V6|M
```

The second alternative is, of course, preferable as it uses up less of the valuable function key buffer.

The second method of hiding the definition from view is implemented in a similar way (the Ctrl code for CLS is |L):

```
*KEY9 -- definition -- |L|M
```

The third method is more complicated.

G CASSETTE ELBUG CASSETTE ELBUG CASSETTE ELBUG CASSETTE ELBUG CASSETTE ELBUG CASSETTE

The January/february edition of the ELBUG magazine cassette is once again packed with useful prpgrams. Many of this month's BEEBUG programs have been included - the Earth from Space, Fuzzy Commands, the complete BEEBUG Filer program, Viewsheet Autoboot, the First Course and Workshop programs, and the demanding Manic Mechanic, a classic platform/arcade game. If that was not all, we have also included the Function Key Lister and Monsters' Gold from your own ELBUG supplement. Fantastic value at just £3.00 plus 50p post & packing.

Once a zero-sized text window has been defined, any further printing to the screen, including the function key definition, is invisible. However, how do we define the text window without that appearing on the screen? The answer is that we use more control codes. Ctrl-\ (|\) is the control code equivalent of the VDU28 - used to define a text window. If we place this code into the function key definition and follow it with four zeros then we will invisibly define a zero-sized window. Of course we cannot simply enter four '0's. These have to be entered as more control codes - Ctrl-@ (|@). A suitable demonstration function key definition might be:

```
*KEY5|\|@|@|@|@P."HELLO EVERYBODY"|M|G|Z|_
|@|J|M
```

The first ten bytes set up the zero-sized window. Then we print a message and sound the beep (|G). Of course, when you use this function key you won't see the message as it is printed inside the zero-sized window. Finally the text window is restored to normality with Ctrl-Z (|Z) and the cursor moved to a position ten lines down (|_|@|A). You can alter the message to a more useful definition and the final resting point of the cursor as you wish.

Your particular choice of the definition-hiding method will depend on the actual definition. Experiment for yourself until you get the result required.

INPUT

A function key can include the INPUT statement to increase the power of the definition.

```
*KEY3 I."Addr:"A:F.I=A TOA+20:P.~I,~?I:N.|
V6|M
```

This definition will prompt for a start address, select mode 6, and then PRINT out the following 20 addresses with their contents.

MACHINE CODE

Another use of the function keys is to call up any machine code program stored independent of a Basic program.

```
*KEY9 CA.&D01|V6|M
```

Will execute any machine code stored starting at address &0D01. If INPUT is used in this context it is possible to pass parameters to the machine code routine from the function key.

BAD KEY

It is likely that at some time you will come across the error message, 'Bad key'. Usually this is issued when you have run out of buffer space. PRINT &0BFF-(&0B00+?&0B10) will give the free space in the buffer at any time. If it is not possible to reduce the definition you are trying to enter by abbreviations or by using control codes, then you have two choices:

1. Remove some of the other key definitions.

2. Remove all the key definitions.

Taking the second choice first, the key buffer is cleared by issuing the command *FX18.

To remove any single key definition, simply type *KEY followed by the number of the key no longer required, and press Return.

Last month we saw how redefining a key by typing in a new definition is implemented. If you examine the buffer regularly, using last month's program, you'll see that the Electron will put the most recent definition last in the buffer, shuffling up the previous definitions as necessary. Resetting the pointers at the same time.

SAVING AND RELOADING DEFINITIONS

Once you have a favourite set of key definitions, you can keep a copy of them on tape using the *SAVE command.

```
*SAVE"KEYS" B00 C00
```

This saves everything stored in the key buffer under the name of KEYS.

To reload, at any time, without affecting any Basic program in memory,

simply use *LOAD.

```
*LOAD"KEYS"
```

This will put the buffer back as it was before.

One tip to those of you familiar with machine code programs. If you store your code in page &A (&0A00 to &0AFF) - the cassette input buffer - then you can save both the machine code routine and the function key buffer in one go with:

```
*SAVE "UTILITY" A00 C00
```

CHECKING DEFINITIONS

If you wish to find out what the function keys are programmed to do, then I offer two suggestions.

1. Type AUTO 10000 (or some other suitable line number not used by your Basic program) and press the appropriate function key. You will see the definition as it is entered into your program (provided no zero-sized window is included in the definition).

2. Run the program listed below. This method has the advantage of showing all the control codes in your definitions and can be used to list key 10 (Break) - not possible with method 1.

```
  10 REM PROGRAM f-KEY LISTER
  20 REM VERSION E0.1
  30 REM AUTHOR  DAVE ROBINSON
  40 REM ELBUG   JAN/FEB 1986
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 MODE6
 110 ON ERROR GOTO 320
 120 REPEAT:REPEAT
 130 INPUT'"Function Key No. ",key%
 140 key%=key% MOD16
 150 end%=FALSE
 160 start%=?(&B00+key%)
 170 IF start%=?&B10 end%=TRUE:PRINT"No
t defined"
 180 UNTIL NOT end%
 190 :
 200 PRINT'"*KEY";key%;" ";
 210 byte%=&B00+start%
 220 REPEAT
 230 byte%=byte%+1
 240 char%=?byte%
 250 IF char%>31 PRINT CHR$(char%);ELSE
PRINT"";CHR$(char%+64);
 260 PROCendcheck
 270 UNTIL end%
 280 PRINT CHR$(13)
 290 UNTIL FALSE
 300 END
 310 :
 320 ON ERROR OFF
 330 IF ERR<>17 REPORT:PRINT" at line "
;ERL
 340 END
 350 :
 360 DEFPROCendcheck
 370 FOR I%=&B00 TO &B0F
 380 IF ?I%=byte%-&B00 end%=TRUE
 390 NEXT I%
 400 ENDPROC
```

FINALLY

In conclusion I list two useful key definitions that will fit into the buffer, together with the previous key 1 and key 10 programs.

The first does essentially the same job as the first program BUFFER, last month, in that it allows RAM memory to be examined.

```
*KEY2 I."Start:"S$:S=EV.S$:F.A=S TOH.
S.8:@%=3:P.~A;:B$=" ";F.B=0TO7:C=A?B:P.~C;
:C=C+(C<320RC>126)*(C-46):B$=B$+CHR$(C):N.
:P.B$:N.|V6|N|M
```

The response to 'Start' can be an address in hex or decimal or an address pointer such as TOP or PAGE.

The complex equation concerning the variable 'C' is to ensure that only printable bytes are displayed. Tokens and control codes appear as full stops (ASCII 46).

My second definition is useful when editing a Basic program. It will list out all line numbers where a specified string is to be found. However, I'm afraid that it won't find Basic keywords.

```
*KEY8I."String:"S$:P=PA.+1:REP.N=256*?P+P?
1:L=P+3:P=P+P?2:IFINS.$L,S$)>0P.N:U.?P=255
:EL.U.?P=255|V6|M
```

Your Guide To BEEBUG

No alterations are required to the programs appropriate to the Electron from this month's BEEBUG. In fact, nearly all the programs in BEEBUG this time will work on an Electron, though a few may need the addition of a Plus 3 or other disc unit. A few general comments on converting Beeb programs to the Electron may also prove useful.

Beeb programs sometimes fail to run on the Electron because of the Beeb's mode 7 and extra sound facilities. However, conversion rarely proves difficult.

The Beeb's mode 7 display is of the same format as mode 6 but with a full range of colours and crude graphics. Mode 7 is not available on the Electron which defaults to mode 6 when it encounters a MODE 7 command. Thus BBC micro programs will often run on an Electron but with a corrupted (and monochrome) display.

The Beeb has three sound channels and control over the volume of the notes. Electron Basic includes all the SOUND and ENVELOPE parameters that are used in the BBC micro even though several have no effect. This means that Beeb programs making full use of sound will still run on the Electron unaltered, though not, of course, producing the full effect.

The other major difference between the BBC micro and the Electron is speed. The Electron is slower, particularly in modes 0, 1 and 2. BBC micro programs often benefit from a change of mode (admittedly, with a loss of some colours) when running on the Electron. Any changes needed or desirable will be given in this ELBUG supplement.

ACORN LAUNCH NEW BBC MICRO

This is the most important announcement for Acorn since the launch of the original BBC micro four years ago. The future of the BBC micro, and thus of Acorn itself, will be just as important for Electron owners. We have all the details in BEEBUG.

THE EARTH FROM SPACE

The results of running this program are extremely attractive views of the Earth and its continents as seen from a point in space. The program will run just as well on the Electron, albeit a little more slowly.

MEMORY EXPANSION BOARDS REVIEWED

As with many hardware add-ons, these boards are designed to work only with the BBC micro.

INTER-SHEET AND INTER-CHART

As with all other Computer Concepts products, these two will only work on the BBC micro. Electron owners, however, can run the excellent Viewsheet spreadsheet package from Acornsoft.

FUZZY COMMANDS

No doubt Electron owners can be just as fumble fingered as those with Beebs. Alan Webster's routine, and the principles on which it is based, will be of just as much interest to ELBUG readers. The program will also work equally well with no alterations necessary.

BEEBUG FILER PART 3

This is the concluding part of this series on developing a database management program. It will work just as well on the Electron for those who have a Plus 3 or other disc unit attached. The complete Filer program is also contained on this month's ELBUG cassette.

DISC RECOVERY

This is another disc based program, but this time it will not work on the Electron, at least with the Plus 3 as this has a quite different file structure.

VIEWSHEET AUTOBOOT

Acornsoft's Viewsheet is available for the Electron (as already mentioned) and this short routine will also work just as well, although it is also dependent upon some type of disc unit being available.

FIRST COURSE - THE EVAL FUNCTION

The EVAL function is part of BBC Basic as used on the Electron. The explanation of its use is therefore just as relevant to Electron owners, and both example programs will run happily on the Electron with no changes.

GAMES REVIEWS

There is a separate Electron version of Repton 2, a really good game, while Software Invasion's Stairway to Hell will work just as well on both the Beeb and the Electron.

WORKSHOP - FLEXIBLE GRAPHS

The versatile procedure described in this workshop for matching up the drawing of axes and graphs to the data available will work on the Electron without the need for any modifications. There is also an extended demonstration program on the ELBUG cassette.

WRITING YOUR OWN COMPILER - PART 2

Contrary to our report in the November supplement, the compiler program as listed will not work on the Electron as the Basic routines called reside at different locations. Given that, the same principles of compiler construction apply to the Electron as to the Beeb.

ADVENTURE GAMES

Unfortunately this adventure game is not available to Electron owners.

HOME FINANCE

All three packages reviewed here are intended for use with disc systems, and all make extensive use of mode seven screens. They are therefore unlikely to run very satisfactorily on the Electron.

MANIC MECHANIC

This excellent game will work with no problems on the Electron, though it is slower in action because of the use of mode 2. Despite that we still believe this makes an interesting and colourful game for Electron owners.

HINTS

There are fewer BEEBUG hints this month and just one is applicable to Electron users:

SINGLE KEY VERIFY

Elbug High Scores

Supplier	Game	Score	Player
ACORNSOFT	Arcadians	11500	A.Gould
ACORNSOFT	Boxer	115590	A.Gould
ELBUG MAG	Builder Bob	372	D.Devlin
ELBUG MAG	Galactic Invasion	86900	K.Bishop
ELBUG MAG	Scrumpy	325370	A.Gould
ELBUG MAG	Spider Man	900	K.Bishop
MICRO POWER	Invaders	12690	S.Bishop
ROMIK	Atom Smasher	2248	P.Seward
SOFTWARE INV	Gunsmoke	4650	R.Cartledge
STARCADE	Savage Pond	19725	R.Cartledge

ACK ISSUES BACK ISSUES BACK ISSUES BACK ISSUES BACK ISSUES BACK ISSUES BACK ISSUE

We are pleased to announce that back issues of the ELBUG magazine, magazine binders and the ELBUG magazine cassette are now available at a discounted price especially for ELBUG members.

ELBUG magazine	Vol.1 and Vol.2	50p per issue
ELBUG magazine binder	any volume	£2.50 each
ELBUG magazine cassette	Vol.1 and Vol.2	£2.00 per cassette

Postage is at the normal rates - For binders and cassettes: 50p for the first item and 30p for each subsequent item. For magazines: 30p for the first item and 10p for each subsequent item. Complete sets of volume 1 are available at £5.00 including postage. Purchasers of magazine back issues from Vol.2 No.4 onwards will receive both the ELBUG supplement for that issue and the corresponding issue of BEEBUG.

Please send all your orders to:
BEEBUG, PO Box 109, St. John's Road, High Wycombe, HP10 8NP.

MONSTERS' GOLD

Andrew Logan describes his all-action arcade game for the nimble-fingered Electron owner.

It is a while since we have featured a game specifically for the Electron in these pages. Monsters' Gold is a fast action game requiring both quick wits and fast fingers.

You control the intrepid adventurer in search of the monster's gold. This is situated at the top of a system consisting of several platforms and ladders and a constantly moving lift. Also dotted around the platforms are a number of sections of ladder that will enable the adventurer to climb to the top platform and collect the gold. So far, so easy. Unfortunately the platforms and ladders are patrolled by a particularly nasty bunch of monsters who only have to collide with the adventurer to lose him one of his three lives. The only place that the adventurer is safe from the monsters is on the right hand platforms and on the lift.

All the time a bonus score is counting down and the gold must be reached before this reaches zero. Once all the sections of the ladder have been collected and returned to the home position the gold may be collected and the next screen started.

The adventurer is controlled with the usual keys - 'Z' and 'X' for left and right. ':' and '/' for up and down the ladders. Type in the program carefully and save it before you run it. Have fun but mind those monsters!

```
  10 REM PROGRAM MONSTERS' GOLD
  20 REM VERSION E0.1
  30 REM AUTHOR  ANDREW LOGAN
  40 REM ELBUG   JAN/FEB 1986
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
  70 ON ERROR GOTO2680
  80 :
 100 MODE1:PROCtitle:MODE5
 110 PROCstart
 120 S%=0:U%=1:H%=3:REM SET SCORE,SHEET
,NUMBER OF LIVES
 130 PROCinit
 140 PROCplatform
 150 PROCladder
 160 PROCsu
 170 REPEAT
 180 PROC1:PROCg:PROCm:PROCscore
 190 UNTILDEAD% OR WON% OR VIC%
 200 IFEX%<=0 GOTO240
 210 IFDEAD% ANDH%<>0 PROCres:GOTO160
 220 IFWON% PROCres:PROCla:GOTO160
 230 IFVIC% PROCvic:PROCdelay(100):PROC
res:CLEAR:CLS:U%=U%+1:GOTO130
 240 CLS:IFS%>=A% A%=S%
 250 COLOUR2:FORI%=0 TO19:PRINTTAB(I%,4
);"*";TAB(I%,28);"*";TAB(I%,13);"*":NEXT
I%
 260 FORI%=4TO28:PRINTTAB(0,I%);"*";TAB
(19,I%);"*":NEXTI%
 270 COLOUR3:PRINTTAB(6,7);"MONSTERS";T
AB(8,10);"GOLD"
 280 COLOUR1:PRINTTAB(5,16);"SCORE ";S%
 290 COLOUR3:PRINTTAB(3,20);"HI-SCORE "
;A%
 300 COLOUR1:PRINTTAB(4,25);"ANOTHER GA
ME?"
 310 REPEAT:I$=GET$:UNTILI$="N"ORI$="Y"
 320 IFI$="Y" CLEAR:CLS:GOTO120
 330 MODE6:END
 340 :
1000 DEFPROCtitle
1010 COLOUR2
1020 PRINTTAB(8,2);"M O N S T E R S  G
O L D":COLOUR1:PRINTTAB(8,3);STRING$(24,
"*")
1030 COLOUR3
1040 PRINT'TAB(1);"Collect the small pi
eces of ladder"''"which hang from the pl
atforms, avoid"''"the monsters and head
back to your"''"starting point(the BLUE
BLOCK) to"''"construct a ladder which wi
ll enable"
1050 PRINT''"you to collect the gold on
the top"''"platform. You must complete t
his task"''"before the bonus runs out if
 you are to"''"continue.  CONTROLS:-"
1060 COLOUR2
1070 PRINT''TAB(7);"'Z'...LEFT"SPC(4)"'
X'...RIGHT"''TAB(7);"':'...UP"SPC(6)"'/'
...DOWN"
1080 COLOUR1:PRINT'''TAB(10);"PRESS SPA
CE TO BEGIN"
1090 REPEAT UNTIL INKEY-99
1100 ENDPROC
```

```
1110 DEF PROCinit
1120 DIMA%(20,31),E%(5),F%(5),M$(3)
1130 DEAD%=FALSE:WON%=FALSE:VIC%=FALSE:
R%=0:V%=0:CC%=30:B%=16:C%=14:UP%=-1
1140 BONUS%=250-(U%*5)
1150 TI%=0
1160 FORI%=1 TO 3:F$(I%)=CHR$(I%+232):N
EXTI%
1170 L$=CHR$231:P$=CHR$230:E$=CHR$232
1180 SL$=CHR$237:BL$=CHR$238:MD$=CHR$23
9
1190 G$=CHR$236:BG$=CHR$240
1200 ENDPROC
1210 DEFPROCstart
1220 VDU23,1,0;0;0;0;
1230 VDU19,1,6,0,0,0
1240 VDU23,230,255,24,36,36,66,66,129,2
55
1250 VDU23,231,129,129,129,255,129,129,
129,255
1260 VDU23,232,255,255,0,0,0,0,0,0
1270 VDU23,233,24,24,0,124,190,25,36,34
1280 VDU23,234,24,24,0,62,125,152,36,68
1290 VDU23,235,90,90,66,126,126,36,36,3
6
1300 VDU23,236,129,66,36,126,90,255,129
,255
1310 VDU23,237,0,0,17,17,31,17,17,0
1320 VDU23,238,255,255,255,255,255,255,
255,255
1330 VDU23,239,0,0,0,0,8,139,139,255
1340 VDU23,240,0,66,36,60,126,60,24,0
1350 A%=0:REM SET HI-SCORE
1360 ENVELOPE1,1,4,-6,-50,206,125,105,1
26,0,0,-126,126,126
1370 ENVELOPE3,1,0,0,0,0,0,0,126,0,0,-1
26,126,126
1380 ENVELOPE4,1,30,0,-30,6,1,6,13,0,0,
-126,126,0
1390 ENDPROC
1400 DEF PROCplatform
1410 COLOUR2
1420 FORI%=14TO 19:PRINTTAB(I%,3);P$;:A
%(I%,3)=-1:NEXTI%
1430  FORJ%=6TO 30 STEP4
1440 FOR I%=0TO 19
1450 IFI%<>16 PRINTTAB(I%,J%);P$;:A%(I%
,J%)=-1
1460 NEXT,
1470 COLOUR1:PRINTTAB(18,30);BL$;TAB(17
,3);BL$
1480 COLOUR2:PRINTTAB(17,2);BG$
1490 ENDPROC
1500 DEF PROCladder
1510 C%=0
1520 COLOUR1
1530 FOR J%=5 TO 25 STEP4
1540 C%=C%+1
1550 IFC%=1 O%=5:P%=10 ELSEIFC%=2 C%=0:
O%=2:P%=13
1560 FOR L%=J% TO J%+4
1570 PRINTTAB(O%,L%);L$;:A%(O%,L%)=2:PR
INTTAB(P%,L%);L$;:A%(P%,L%)=2
1580 NEXT,:ENDPROC
1590 DEFPROCm
1600 IFDEAD% ORWON% ORVIC% ENDPROC
1610 N%=X%:M%=Y%:V%=Z%
1620 IFX%=16 GOTO1650
1630 IFINKEY-73 Z%=3:Y%=Y%-1:STILL%=0:G
OTO1690
1640 IFINKEY-105 Z%=3:Y%=Y%+1:STILL%=0:
GOTO1690
1650 IFINKEY-98 Z%=1:X%=X%-1:STILL%=0:G
OTO1680
1660 IFINKEY-67 Z%=2:X%=X%+1:STILL%=0:G
OTO1680
1670 STILL%=TRUE:GOTO1690
1680 IFX%=-1 GOTO1690 ELSEIFA%(X%,Y%+1)
=-1 SOUND1,4,14,1
1690 IFX%=-1 X%=0
1700 IFX%=20 X%=19
1710 IFML% ANDX%=0 X%=1
1720 IF(Z%=3 ANDA%(X%,Y%)<>2)OR(Z%<3 AN
DA%(X%,Y%+1)=0)X%=N%:Y%=M%:Z%=V%
1730 IFML%GOTO1750
1740 IFX%=LX% ANDY%=LY%+2 ML%=-1:S%=S%+
10:PRINTTAB(LX%,LY%);SPC1
1750 IFX%=16 Y%=C%-1
1760 IFSTILL% ANDX%<>16 ENDPROC
1770 IFA%(N%,M%)=2 COLOUR1:PRINTTAB(N%,
M%);L$ ELSEPRINTTAB(N%,M%);SPC1
1780 IFNOTML% GOTO1800
1790 IFA%(N%-1,M%)=2 COLOUR1:PRINTTAB(N
%-1,M%);L$ ELSEIFA%(N%-1,M%)=-1 COLOUR2:
PRINTTAB(N%-1,M%);P$ ELSEPRINTTAB(N%-1,M
%);SPC1
1800 COLOUR3:PRINTTAB(X%,Y%);M$(Z%)
1810 IFML% COLOUR1:PRINTTAB(X%-1,Y%);SL
$
1820 IFY%=2 ANDX%=17 VIC%=-1:S%=S%+(EX%
*50):ENDPROC
1830 IFA%(X%,Y%)=4 ORA%(X%,Y%)=5 PROCde
ad
1840 IFML% ANDX%<>16:IFA%(X%-1,Y%)=4 OR
A%(X%-1,Y%)=5 PROCdead
1850 IFML% ANDX%=18 ANDY%=29 THENWON%=-
1:TI%=TIME
1860 ENDPROC
1870 DEFPROC1
1880 Q%=B%:W%=C%
1890 IFUP% C%=C%-1 ELSEIFUP%=0 C%=C%+1
1900 IFC%=31 UP%=-1:ENDPROC
1910 IFC%<6 UP%=0:ENDPROC
1920 PROCm
1930 IFUP% PRINTTAB(Q%,W%);SPC1; ELSEIF
X%<>16 PRINTTAB(Q%,W%);SPC1;
1940 COLOUR2:PRINTTAB(B%,C%);E$;
1950 A%(B%,C%)=-1:A%(Q%,W%)=0
1960 ENDPROC
1970 DEFPROCg
```

```
 1980 IFDEAD% ENDPROC
 1990 R%=R%+1:IFR%>4 R%=1
 2000 K%=E%(R%):L%=F%(R%)
 2010 IFA%(K%,L%+SGN(Y%-L%))=2 F%(R%)=F%
(R%)+SGN(Y%-L%):GOTO2030
 2020 IF(L%-1)/4=(L%-1)DIV4 E%(R%)=E%(R%
)+SGN(X%-K%)
 2030 IFA%(E%(R%),F%(R%))=4 ORA%(E%(R%),
F%(R%))=5 ORE%(R%)=16 ORE%(R%)=-1 PROChy
p:SOUND0,-15,200,1:GOTO2050
 2040 SOUND1,-15,10,1
 2050 IFA%(E%(R%),F%(R%))=0 A%(E%(R%),F%
(R%))=4
 2060 IFA%(E%(R%),F%(R%))=2 A%(E%(R%),F%
(R%))=5
 2070 IFA%(K%,L%)=4 PRINTTAB(K%,L%);SPC1
:A%(K%,L%)=0
 2080 IFA%(K%,L%)=5:COLOUR1:PRINTTAB(K%,
L%);L$:A%(K%,L%)=2
 2090 COLOUR3:PRINTTAB(E%(R%),F%(R%));G$
 2100 IFX%=E%(R%)ANDY%=F%(R%)PROCdead
 2110 IFML% ANDX%<>16 ANDE%(R%)=X%-1 AND
F%(R%)=Y% PROCdead
 2120 ENDPROC
 2130 DEFPROChyp
 2140 REPEAT:E%(R%)=RND(14):F%(R%)=(RND(
7)*4)+1:UNTILE%(R%)<>X% ANDE%(R%)<>X%-1
AND(A%(E%(R%),F%(R%))=0 ORA%(E%(R%),F%(R
%))=2)
 2150 ENDPROC
 2160 DEFPROCsu
 2170 UP%=-1:DEAD%=0
 2180 ML%=0:WON%=0
 2190 X%=18:Y%=29:Z%=1
 2200 COLOUR3:PRINTTAB(X%,Y%);M$(1)
 2210  FORI%=1 TO4:E%(I%)=8:NEXTI%
 2220 F%(1)=5:F%(2)=13:F%(3)=25:F%(4)=21
 2230 FORI%=1TO4:PRINTTAB(E%(I%),F%(I%))
;G$:A%(E%(I%),F%(I%))=4:NEXTI%
 2240 REPEAT:LX%=RND(12):LY%=(RND(6)*4)+
3:UNTILA%(LX%,LY%)=0:COLOUR1:PRINTTAB(LX
%,LY%);SL$
 2250 PRINTTAB(6,31);"SHEET ";U%;
 2260 COLOUR3:FORI%=1TOH%:PRINTTAB(I%,31
)M$(1);:NEXTI%
 2270 PROCdelay(150):TIME=TI%
 2280 ENDPROC
 2290 DEFPROCres
 2300 FORI%=1TO4:IF A%(E%(I%),F%(I%))=4
PRINTTAB(E%(I%),F%(I%));SPC1:A%(E%(I%),F
%(I%))=0
 2310 IF A%(E%(I%),F%(I%))=5:COLOUR1:PRI
NTTAB(E%(I%),F%(I%));L$:A%(E%(I%),F%(I%)
)=2
 2320 NEXTI%:IFA%(X%,Y%)=2:COLOUR1:PRINT
TAB(X%,Y%);L$ ELSEPRINTTAB(X%,Y%);SPC1
 2330 IFML%:IFA%(X%-1,Y%)=2 COLOUR1:PRIN
TTAB(X%-1,Y%);L$ ELSEIFA%(X%-1,Y%)=-1 CO
LOUR2:PRINTTAB(X%-1,Y%);P$ ELSEIFA%(X%-1
,Y%)=0 PRINTTAB(X%-1,Y%);SPC1
 2340 PRINTTAB(B%,C%);SPC1;:A%(B%,C%)=0
 2350 PRINTTAB(LX%,LY%);SPC1
 2360 ENDPROC
 2370 DEFPROCdead:TI%=TIME:PRINTTAB(H%,3
1);SPC1;:H%=H%-1:DEAD%=-1
 2380 *FX15,0
 2390 COLOUR3:PROCdelay(25):SOUND0,3,5,1
 2400 IFML% PRINTTAB(X%-1,Y%);" "+MD$ EL
SEPRINTTAB(X%,Y%);MD$
 2410 PROCdelay(100):ENDPROC
 2420 DEFPROCla
 2430 CC%=CC%-4:IFCC%<1 ENDPROC
 2440 SOUND1,1,67,10
 2450 FORI%=CC% TOCC%+3
 2460 COLOUR1:PRINTTAB(19,I%);L$;:A%(19,
I%)=2
 2470 NEXTI%
 2480 S%=S%+100:PROCdelay(100)
 2490 ENDPROC
 2500 DEF PROCdelay(DEL%):NOW%=TIME:REPE
AT:UNTIL TIME>NOW%+DEL%:ENDPROC
 2510 DEFPROCscore
 2520 IFDEAD%ORWON% ENDPROC
 2530 COLOUR3:PRINTTAB(0,3);"SCORE ";S%
 2540 EX%=BONUS%-(INT(TIME/100))
 2550 COLOUR1:PRINTTAB(6,1);"BONUS ";EX%
 2560 IFEX%<=99 PRINTTAB(14,1);SPC1
 2570 IFEX%<=9 PRINTTAB(13,1);SPC1
 2580 IFEX%<=0 PROCdead
 2590 ENDPROC
 2600 DEFPROCvic:PROCdelay(75)
 2610 COLOUR2:PRINTTAB(17,2);BG$:COLOUR3
:PRINTTAB(18,2);M$(2):SOUND1,4,14,1:PROC
delay(25):COLOUR2:PRINTTAB(17,2);SPC1;TA
B(18,2);BG$:COLOUR3:PRINTTAB(19,2);M$(2)
:SOUND1,4,14,1:PROCdelay(25):PRINTTAB(18
,2);SPC1
 2620 FORI%=3TO28:COLOUR1:PRINTTAB(19,I%
-1);L$;:COLOUR2:PRINTTAB(19,I%);BG$;:COL
OUR3:PRINTTAB(19,I%+1);M$(3);:PROCdelay(
15):NEXTI%
 2630 COLOUR1:PRINTTAB(19,28);L$;TAB(19,
29);L$
 2640 COLOUR2:PRINTTAB(17,29);BG$:COLOUR
3:PRINTTAB(18,29);M$(2):SOUND1,4,14,1:PR
OCdelay(40):PRINTTAB(17,29);SPC1
 2650 FORI%=1TO12:COLOUR2:PRINTTAB(18,27
);BG$:COLOUR3:PRINTTAB(18,28);M$(3):PRIN
TTAB(18,29);SPC1:PROCdelay(15):PRINTTAB(
18,27);SPC1:COLOUR3:PRINTTAB(18,29);M$(3
):COLOUR2:PRINTTAB(18,28);BG$:SOUND1,4,1
4,1:PROCdelay(15):NEXTI%:PROCdelay(70)
 2660 ENDPROC
 2670 ON ERROR OFF
 2680 MODE6
 2690 IF ERR<>17 REPORT:PRINT" at line "
;ERL
 2700 END
```